

Early Chronic illness Detection

Satyam (2021284)

Aditya Jain (2021305)

Suyash Kumar (2021293)

Vickey Kumar (2021299)



INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI



Addressing Rising Chronic illness Rates

- Globally, An estimated 3.8% of the population experience depression, including 5% of adults, and 5.7% of adults older than 60 years. [1]
- India's **suicide rate** is among the highest globally, with an estimated **21.1 suicides per 100,000 people**, indicating the severe impact of untreated mental health issue.[2]

The Impact of Mental Health Issues

- Depression is a leading cause of disability worldwide, contributing significantly to the global burden of disease.
- The **Indian Council of Medical Research (ICMR)** highlighted that mental health disorders, including depression, are pushing around **20% of Indian households** into poverty due to high healthcare costs.

Importance of Timely Detection

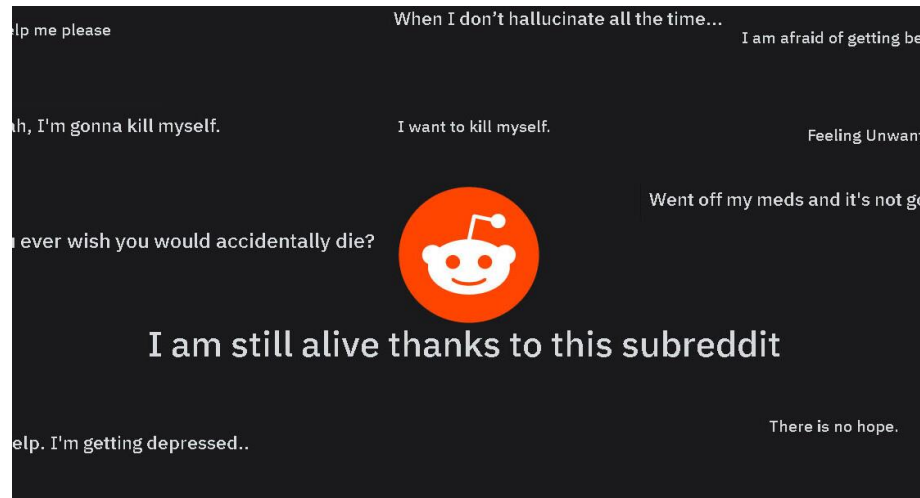
- **Prevention of Severe Outcomes:** Timely detection of depression can reduce the risk of severe mental health conditions, such as chronic depression or suicide
- **Cost-Effective Treatment:** Early identification allows for less intensive and more cost-effective interventions, reducing healthcare expenses and improving long-term outcomes for individuals [3]

Motivation



How we use in Real Life

- We have created a model that collects individual data through a Google form by collaboration with college wellbeing cells. Google forms covering questions like lifestyle and family history, to predict chronic health conditions.
- Based on the predictions, we provide personalized health recommendations and share doctor contact information for early intervention.



Depression Detection Using ML (Chauhan et al. 2023):

- Establishes a relationship between stress-related factors and depression detection using machine learning techniques.
- Stress related factors are qualitatively.
- This study focuses on identifying individuals who may not outwardly show signs of depression but are under stress.

Neuroimaging-Based Detection (Fu et al. 2018):

- ML analysis of neuroimaging data identified structural and functional brain changes related to depression, with an accuracy of 86%.
- This method use CV to analyze neuroimaging based detection

Speech and Language Patterns for Depression Detection (Alghowinem et al. 2016):

- ML algorithms analyze speech features like slower speech rate, longer pauses, and lower pitch, achieving an accuracy of 80% in predicting depression.
- The SVM model and NLP was effective in identifying distinct speech patterns linked to depressive symptoms

Concluding Remarks:

- Our project focus on - Integration of Additional Factors: Future research could integrate more comprehensive datasets, including historical markers (e.g., Family History of Depression, Mental health and substance) and daily activity factors(sleep pattern), to enhance the prediction accuracy.

Dataset Overview

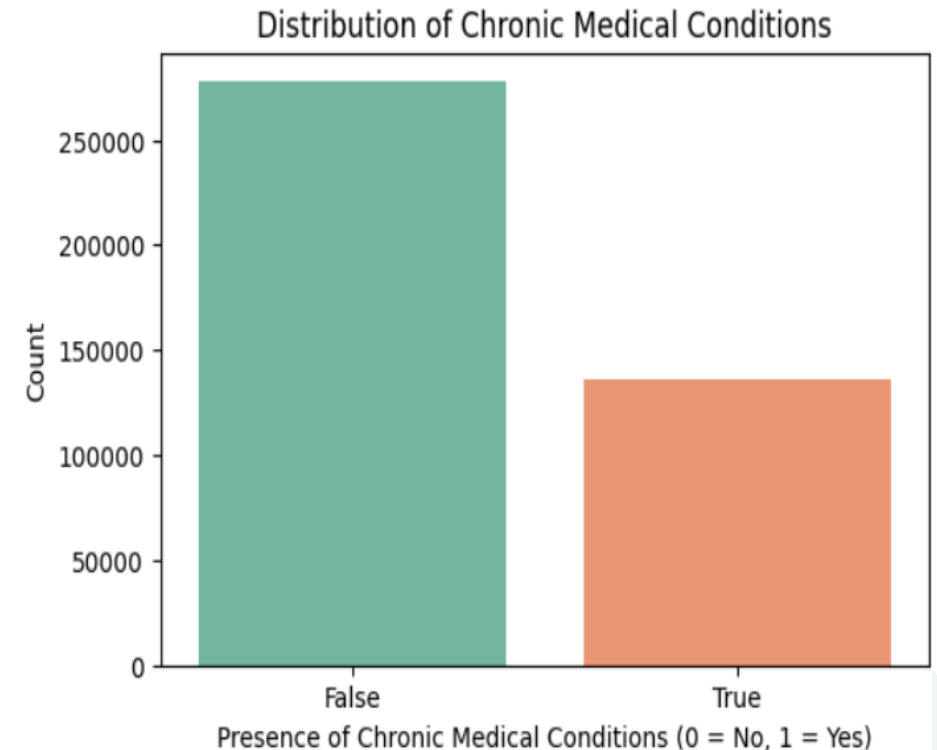


Data Collection:

- Data sourced from Kaggle {**Author - Anthony Therrien**}, i.e [Dataset](#)
- This dataset contains information on individuals with various attributes related to their personal and lifestyle factors. It is designed to facilitate analysis in areas such as health, lifestyle, and socio-economic status..

Demographic of Dataset Entries:

- Data consists of **413768** entries with label 'No' or 'Yes' for Chronic Disease and **16 columns**



Dataset Overview



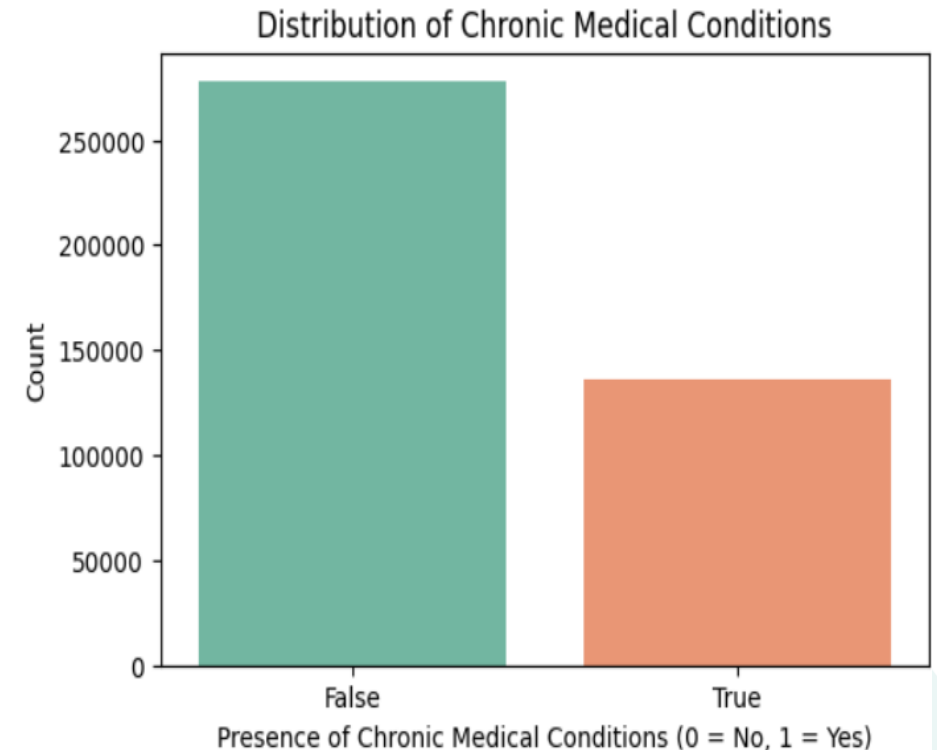
Data Attributes

The dataset contains the following attributes

- **Demographic** factors (e.g., age, marital status, education)
- **Health and lifestyle** indicators (e.g., smoking status, physical activity level, sleep patterns)
- **Medical and Psychological history** (e.g., mental illness, substance abuse, family history of depression)
- **Socio-economic** variables (e.g., income, employment status).

Demographic of Dataset Entries:

- Data consists of **413768** entries with label **‘No’** or **‘Yes’** for Chronic Disease and **16 columns**



Data Preprocessing & EDA



Data Pre-Processing:

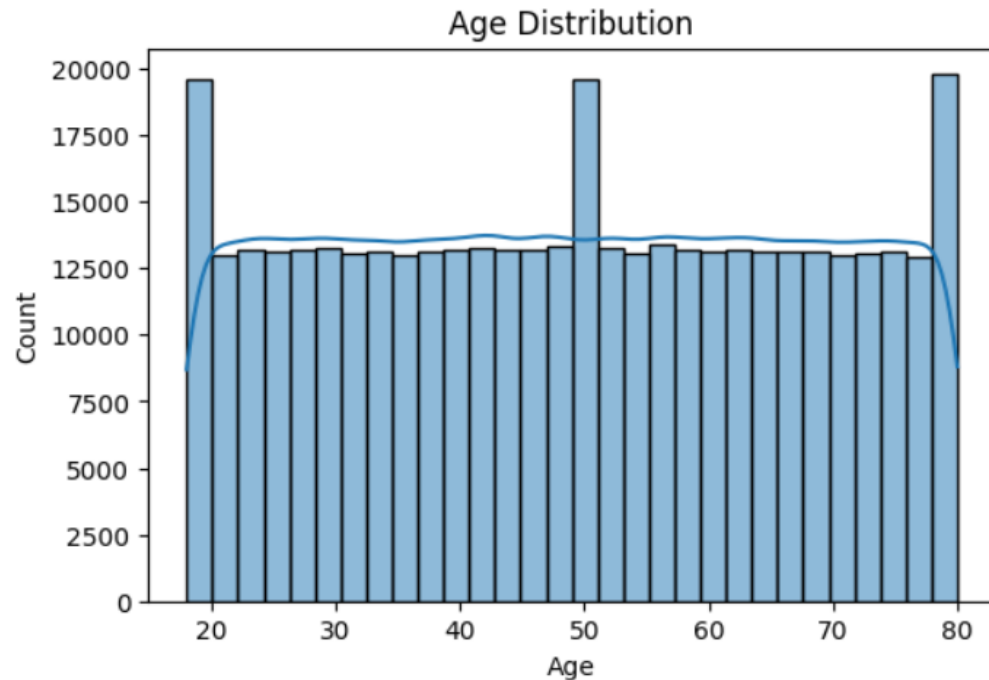
- **Class imbalance** was addressed using SMOTE to balance "No" and "Yes"
- **Standardization** minor tweaking is expected to be scaled
- **One-hot encoding** for categorical variables like Marital Status, Education Level, and Smoking status etc
- **Missing Value**
- **Non- Linear Relationships**



Exploratory Data Analysis:



Data Analysis

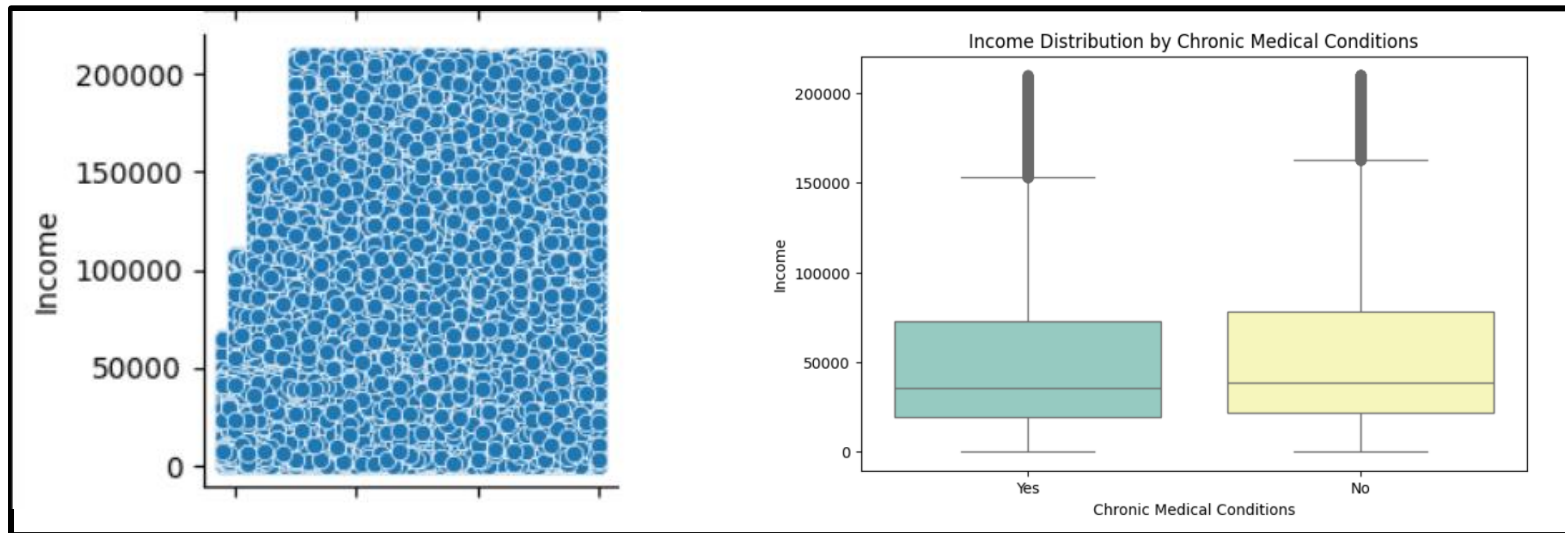


- Age distribution shows peaks at 20, 50, and 80, suggesting targeted survey samples.
- Relationships between age, income, lifestyle, and chronic conditions are likely non-linear in health data..

Exploratory Data Analysis:



Income vs Age & Income vs Chronic medical condition Analysis



- **Income distribution:** Income seems to have a broad spread across all ages, ranging from 0 to over 200,000. There is no strong visual correlation between age and income.
- **Income clustering:** The income distribution is relatively dense across higher age groups, with no significant clustering in lower age groups.

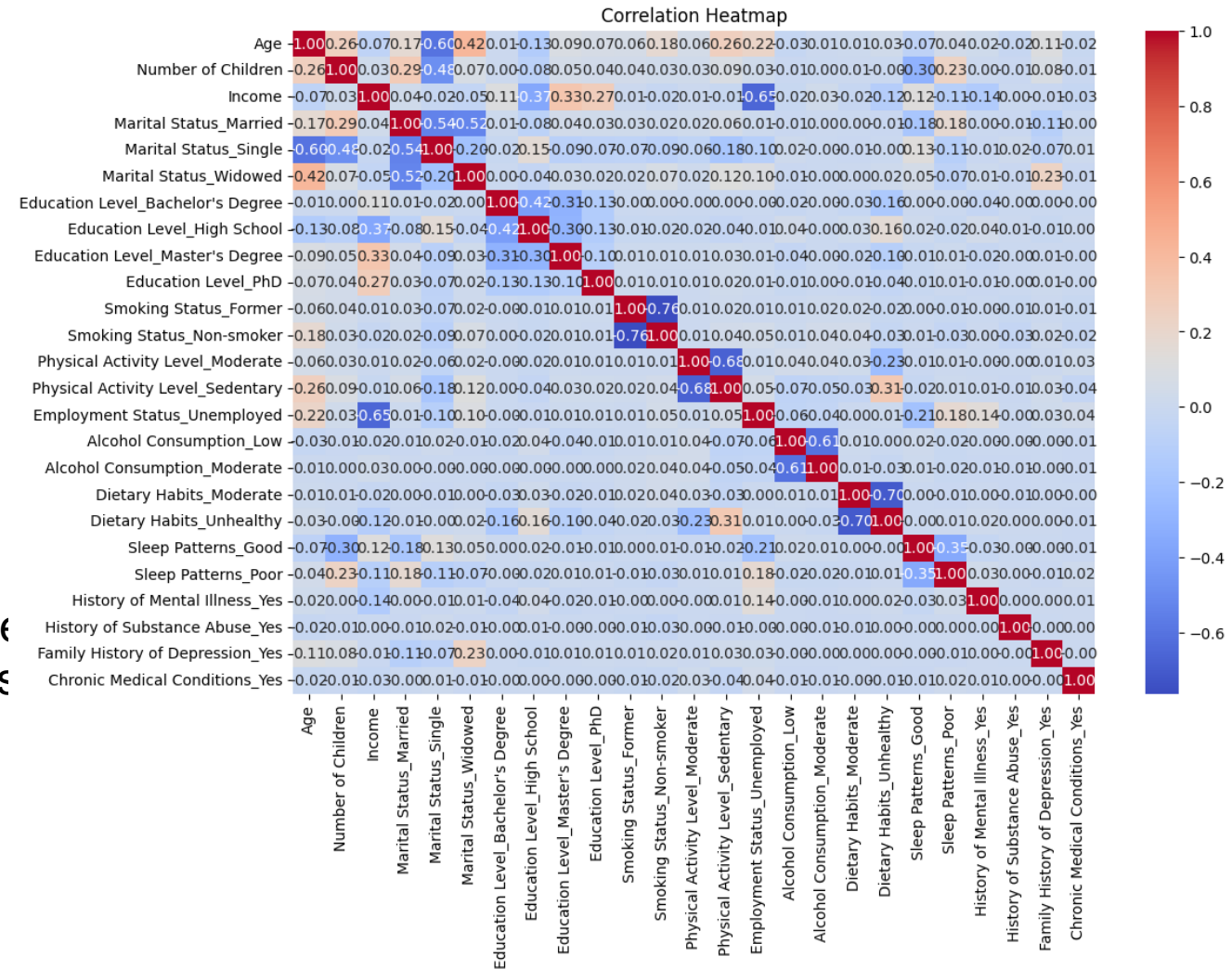
Exploratory Data Analysis:



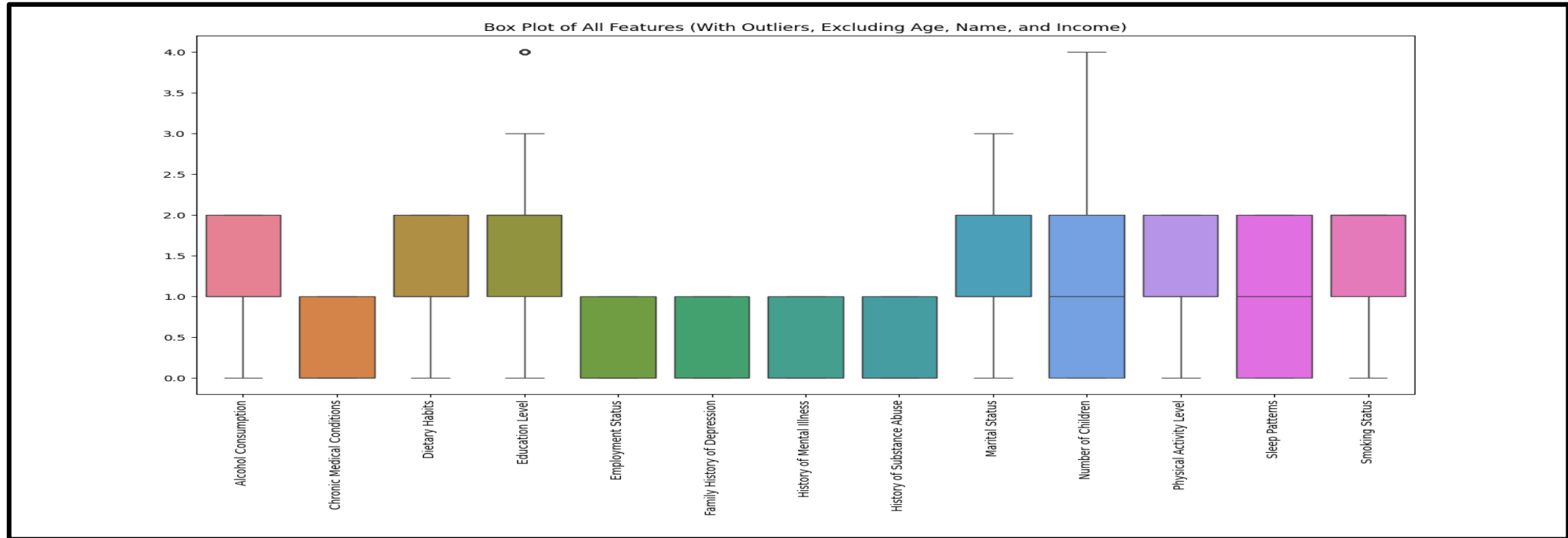
Heat Maps

The heatmap shows correlations between various factors in the dataset.

- **Positive correlations** (in red) indicate that as one factor increases, so does the other, such as income and employment or physical activity and good sleep.
- **Negative correlations** (in blue) show an inverse relationship, where an increase in one factor leads to a decrease in another, such as unemployment with income or poor sleep patterns with physical activity and healthy habits.



Outlier Detection



Boxplot to detect the outliers in the data. we can clearly see that there are many sample points in the dataset that show the central tendency of the data hence they do not display prominent outliers, meaning their data distributions are more contained within the range represented by their whiskers.

Methodology – Logistic Regression

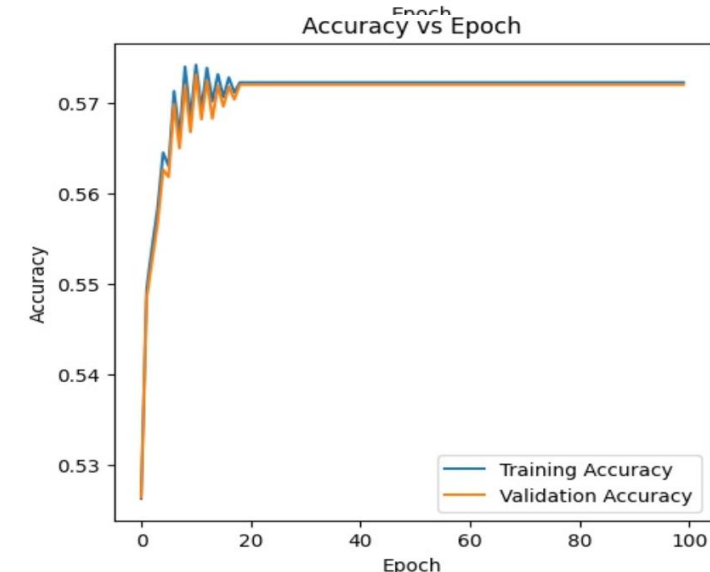
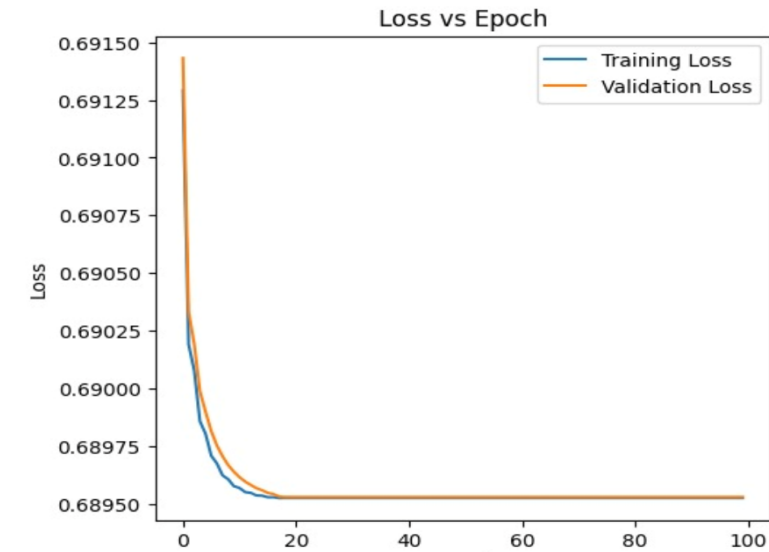


Logistic Regression with Batch Gradient Descent (BGD) – 57%

- BGD ensures more stable updates by computing gradients over the entire dataset, leading to better convergence.
- Handles class imbalance effectively with balanced class weights, contributing to moderate performance.

Logistic Regression with Stochastic Gradient Descent (SGD) – 52%

- SGD updates model parameters with each sample, resulting in noisier updates and potential instability.
- Performance is lower due to sensitivity to learning rate and less precise convergence compared to BGD.



Methodology – Why Batch gradient > SGD ?

Stability of Updates:

- **SGD** is more **sensitive to learning rate** because it updates the weights frequently (one sample at a time). If the learning rate is not tuned correctly, it can result in **poor convergence or oscillations**.
- In **BGD**, the weights are updated after processing the entire dataset, making it **less sensitive to learning rate adjustments**.

BGD achieves **better convergence** since it minimizes the loss more consistently, especially on smaller or well-conditioned datasets.

Sensitivity to Learning Rate:

- **Batch Gradient Descent Logistic Regression** integrates **L2 regularization** by default, which helps prevent overfitting.
- **SGDClassifier** requires additional parameter tuning to achieve optimal regularization. If not properly tuned, it may **overfit or underfit** the data.

BGD can achieve **better results** even with default hyperparameters, while **SGD may require careful tuning** of learning rate or other hyperparameters.

Methodology – Performing RFE to find Best Top Features & L2 Regularization



L2 (Ridge) Logistic Regression with Recursive Feature Elimination (RFE) On top 5 features– 56%

L2 (Ridge) Logistic Regression with Recursive Feature Elimination (RFE) On top 10 – 57%

Selected Feature of RFE : Education Level, Smoking Status, Physical Activity Level, Employment Status, Alcohol Consumption, Dietary Habits ...etc



Methodology – Why RFE + L2 Regularization has better Accuracy??




- RFE selects the most informative features, enhancing performance by reducing dimensionality.
- Ridge regularization prevents overfitting, leading to better generalization.



Purpose of PCA (Principal Component Analysis)

- PCA is a **dimensionality reduction technique** that transforms the original features into a smaller set of components, retaining as much variance (information) as possible. This is useful to simplify models and reduce overfitting.

Logistic Regression with PCA – 54%

- PCA simplifies the data by retaining only the most significant components, reducing noise.
 - However, some relevant feature information might be lost during dimensionality reduction, slightly impacting performance.
- 

Methodology – Logistic Regression with SMOTE and Optimal Threshold :



Purpose of SMOTE

- SMOTE **balanced** the dataset, which ensures that the model will give **equal importance** to both classes, improving performance for the minority class and preventing biased predictions.

Purpose of Threshold

- By **tuning the decision threshold**, we move away from the default 0.5 threshold to one that optimizes the model's performance based on the ROC curve.

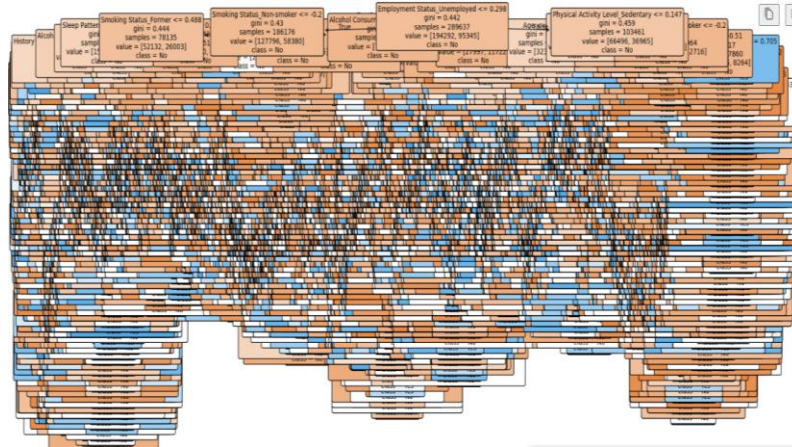
Logistic Regression with SMOTE and Optimal Threshold – 60%



Methodology – Decision Tree



Without
PRUNING



With Pruning

1. Decision Tree (Gini Criterion) – Accuracy 57%

1. Captures non-linear relationships but lacks robustness in handling complex interactions compared to other ensemble methods.

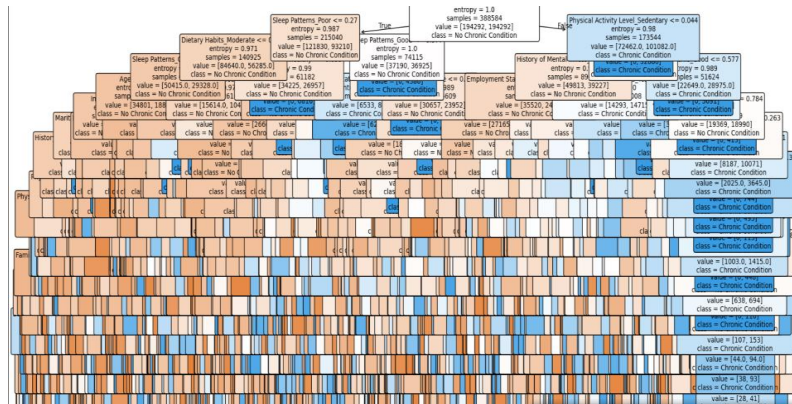
2. Decision Tree (Entropy Criterion) – Accuracy 63%

1. Entropy-based splitting captures finer information gain, improving the model's ability to separate classes better than Gini.

Without Pruning Accuracy : 55%

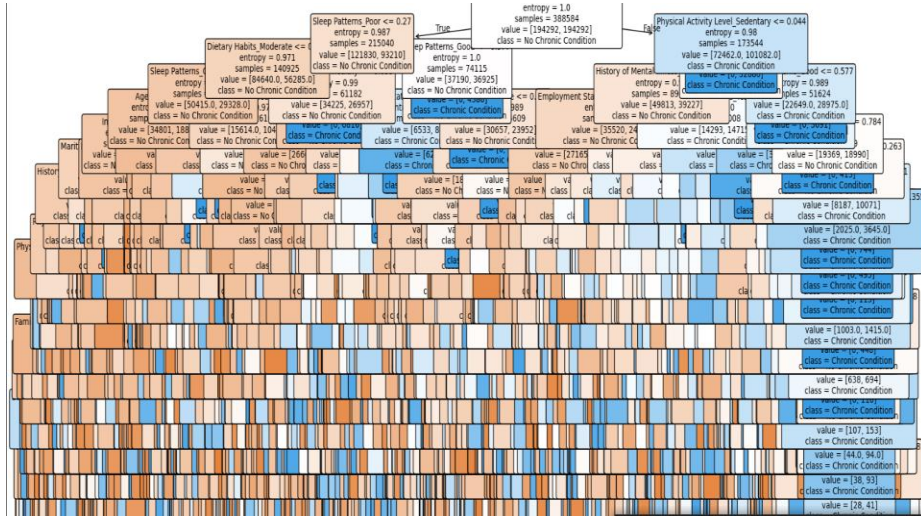
Entropy > Gini

With
PRUNING



t-SNE

Methodology – Decision Tree With Pruning



Perform Hyperparameter Tuning Using **GridSearchCV** + **SMOTE**-Balanced Training Dataset to find the Best Combination of Hyperparameters.

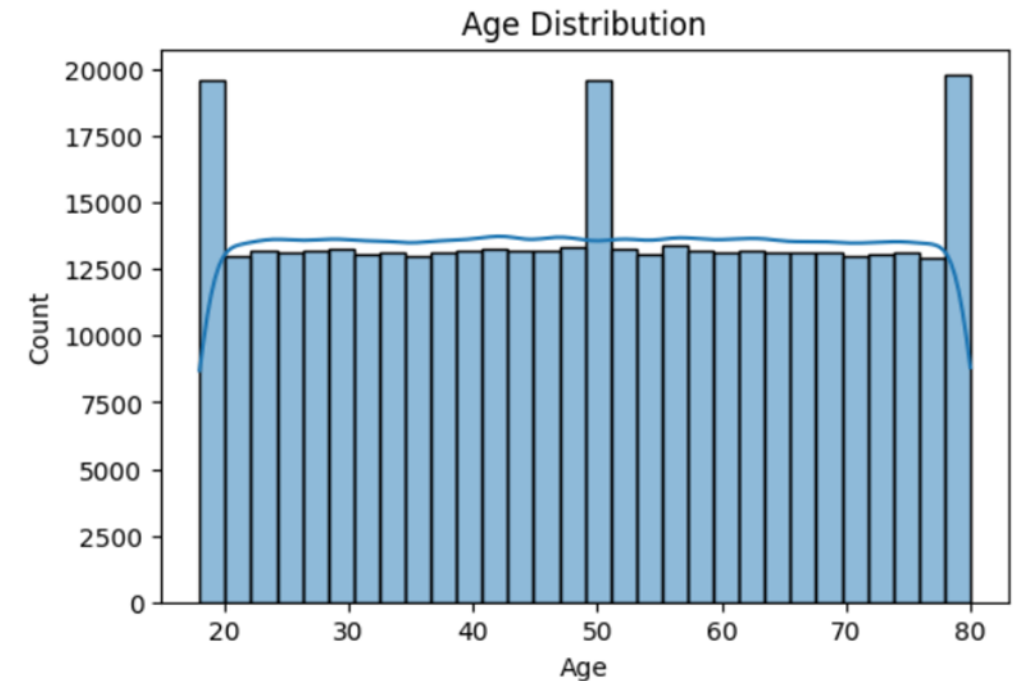
Best Decision Tree Parameters: {'criterion': 'entropy',
'max_depth': 20, 'min_samples_leaf': 2,
'min_samples_split': 5}

Accuracy: 63%

Methodology – Why Decision Tree With Pruning High Accuracy ??



- **Captures Non-Linear Relationships eg**
Age and Income :
- **Class Imbalance Handling via Splitting & SMOTE:**
- **Flexible Hyperparameters:**
- **No Overfitting** due to Flexible Hyperparameter
- **Improved Generalization and Robustness**
- **Handles Feature Interactions:** (via Entropy and Ginni)



Methodology – Naïve Bayes



Bernoulli Naive Bayes – 67%

- Optimized for binary features, it matches Gaussian Naive Bayes in performance.
- SMOTE helps in balancing the class distribution, further improving predictions.



Methodology – Random Forest



1. Random Forest (Default parameters) – 59%

- The default setup captures non-linear patterns but lacks parameter optimization, which limits its full potential.
- Handles feature interactions but suffers from overfitting with deep trees.

1. Random Forest (Tuned) – 61%

- Tuned parameters (max_depth, n_estimators) prevent overfitting and enhance generalization.
- Class weights ensure balanced predictions across minority and majority classes.

Why Random Forest (Tuned) > Random Forest (Default Parameter) ??

Methodology – Why the Tuned Model Performed Better (Accuracy: 61% vs. 59%)



Hyperparameter Tuning: The second model was tuned to avoid **overfitting**, ensuring it **Generalizes better on unseen data**.

Balanced Class Weights: **Improved handling of class imbalance** contributes to better performance, especially for minority classes.

More Trees: Increasing the number of trees to 200 stabilizes predictions by **Reducing variance**.

Depth Control: **Limiting the depth to 20** prevents the trees from becoming too complex, improving generalization.



Data Pre-Processing:

- **Class imbalance** was addressed using SMOTE to balance "No" and "Yes"
- **Standardization** minor tweaking is expected to be scaled
- **One-hot encoding** for categorical variables like Marital Status, Education Level, and Smoking status etc
- **Missing Value**
- **Non- Linear Relationships**
- **Handling Missing or Irrelevant Columns:** The column **Name** is dropped as it is irrelevant to the model
- **Feature and Target Separation:** Features (**X**) and the target (**y**) are separated. The target is **Chronic Medical Conditions_Yes**.

MULTI-LAYER PERCEPTRON(MLP)



Hidden Layers: Two layers with 100 and 50 neurons, respectively, allowing the model to capture complex patterns in the data.

Activation Functions: Defines how neurons fire and introduce non-linearity in the model. Options include:

identity: Linear activation (no transformation).

logistic: Sigmoid function, suitable for probabilistic outputs.

tanh: Hyperbolic tangent, outputs between -1 and 1.

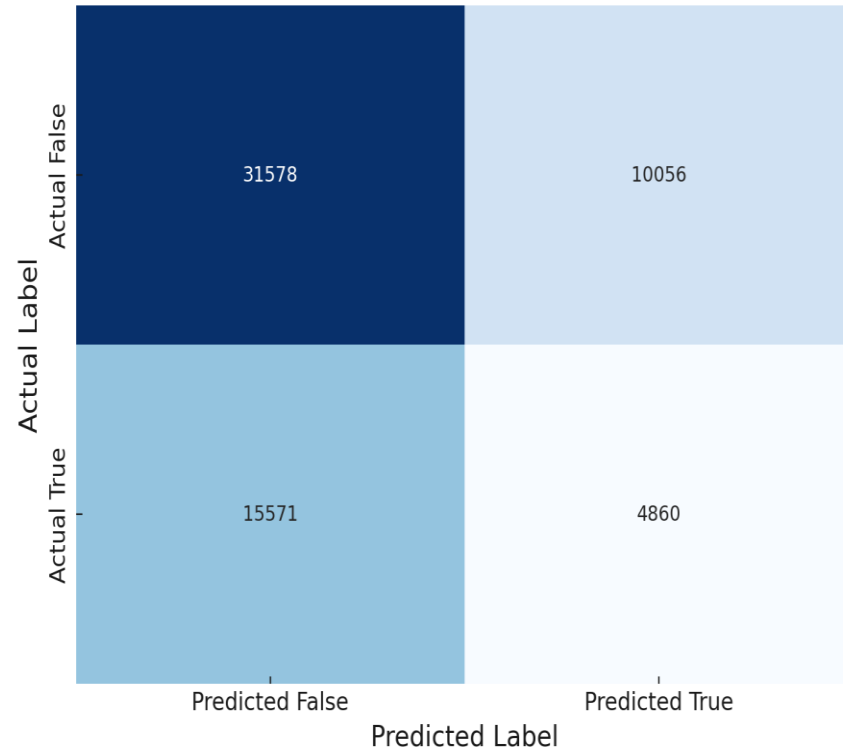
relu: Rectified Linear Unit, introduces sparsity and faster training.

- **Optimizer:** **adam** (adaptive moment estimation), which adjusts learning rates dynamically for faster convergence.

MLP with Identity Activation Function



Confusion Matrix



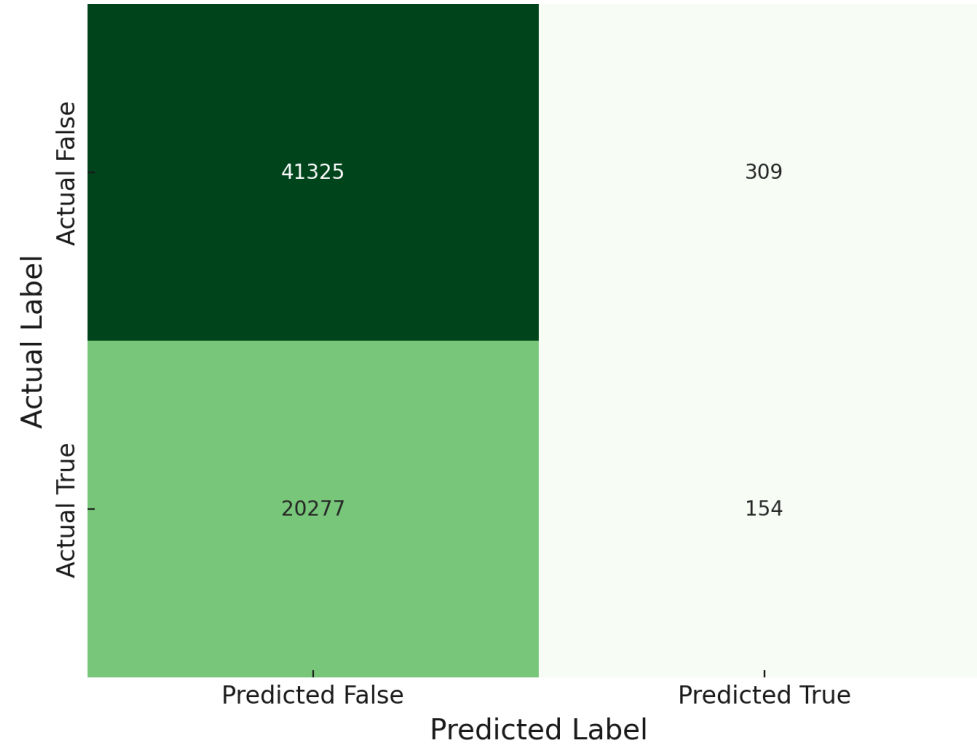
Classification Report

Class	Precision	Recall	F1-Score	Support
False	0.67	0.76	0.71	41634
True	0.33	0.24	0.27	20431
Overall Accuracy	0.59	0.59	-	-
Macro Average	0.5	0.5	0.49	62065
Weighted Average	0.56	0.59	0.57	62065

MLP with Logistic



Confusion Matrix



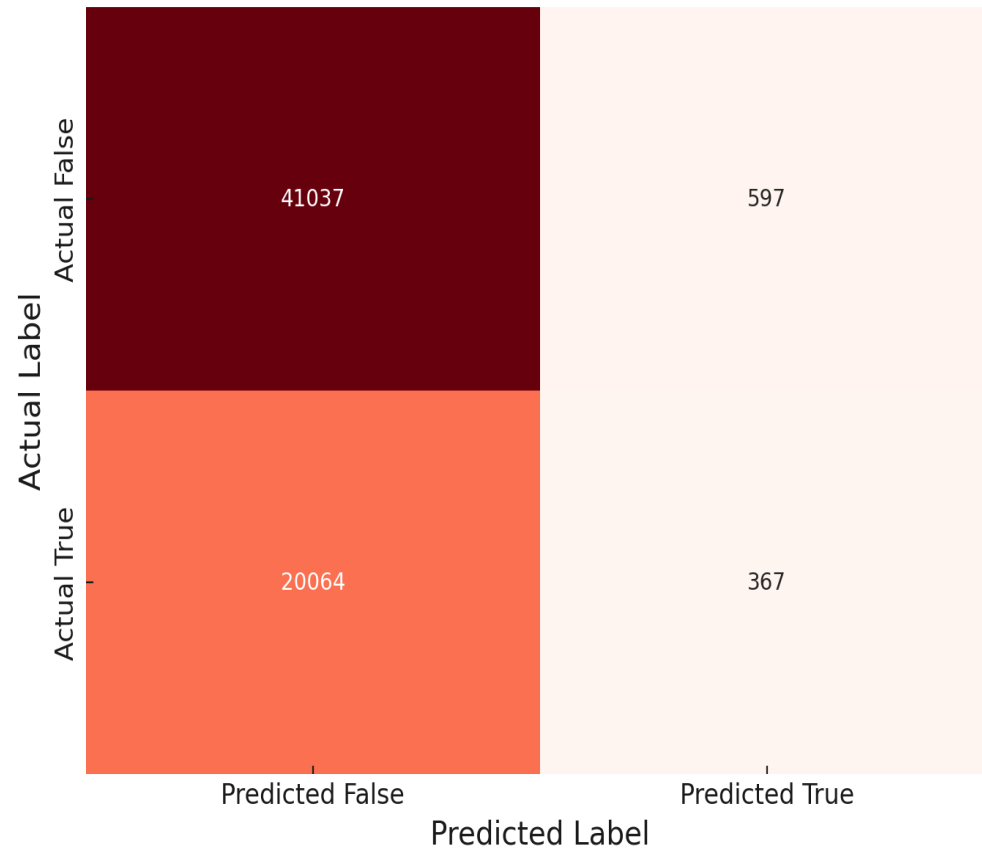
Classification Report

Class	Precision	Recall	F1-Score	Support
False	0.67	0.99	0.8	41634
True	0.33	0.01	0.01	20431
Overall Accuracy	0.67	0.67	-	-
Macro Average	0.5	0.5	0.41	62065
Weighted Average	0.56	0.67	0.54	62065

MLP with tanh



Confusion Matrix



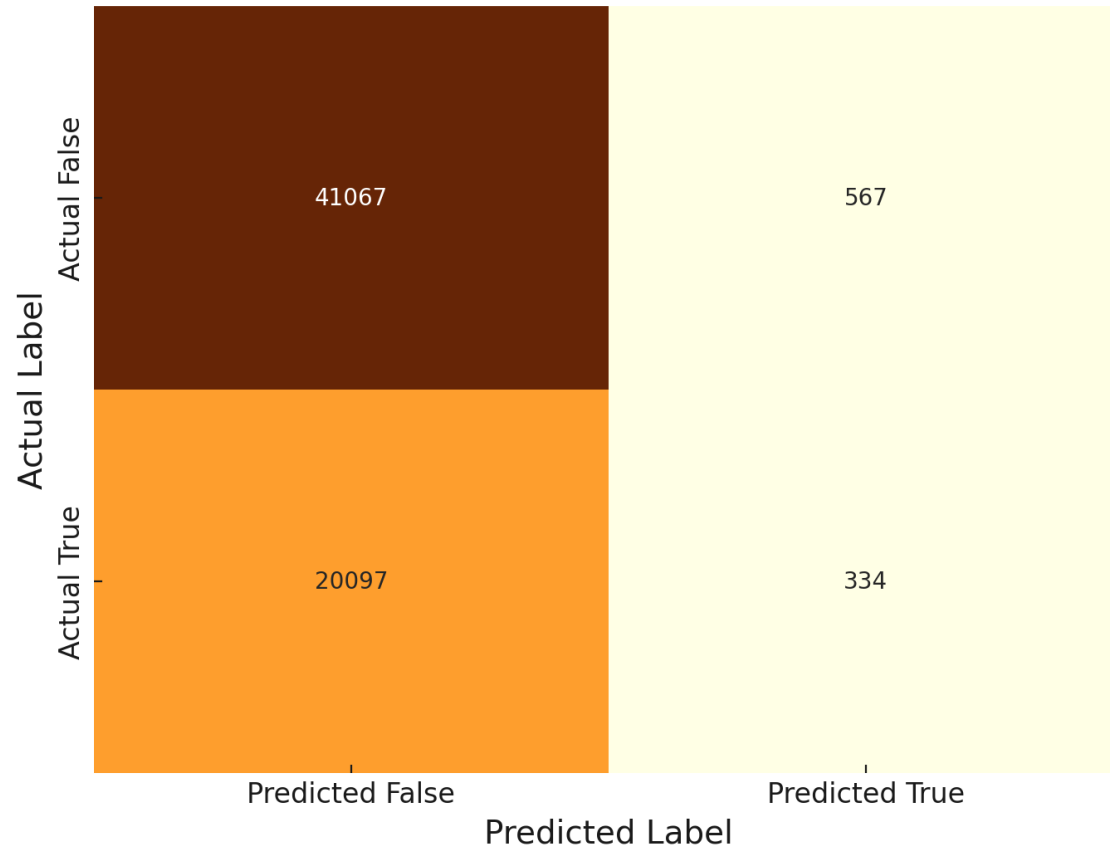
Classification Report

Class	Precision	Recall	F1-Score	Support
False	0.67	0.99	0.8	41634
True	0.38	0.02	0.03	20431
Overall Accuracy	0.67	0.67	-	-
Macro Average	0.53	0.5	0.42	62065
Weighted Average	0.58	0.67	0.55	62065

MLP with ReLu



Confusion Matrix



Classification Report

Class	Precision	Recall	F1-Score	Support
False	0.67	0.99	0.8	41634
True	0.37	0.02	0.03	20431
Overall Accuracy	0.67	0.67	-	-
Macro Average	0.52	0.5	0.42	62065
Weighted Average	0.57	0.67	0.55	62065

MULTI-LAYER PERCEPTRON (MLP)



MLP Classifier('IDENTITY')	58
MLP Classifier('LOGISTIC','TANH','RELU')	66

MLP Classifier with Identity Activation – 58.7%

- Acts as a linear model, limiting the network's ability to learn non-linear patterns effectively.

MULTI-LAYER PERCEPTRON (MLP)



MLP Classifier('IDENTITY')	58
MLP Classifier('LOGISTIC','TANH','RELU')	66

MLP Classifier with Logistic/Tanh/ReLU Activation – 66-67%

- Non-linear activations (Logistic, Tanh, ReLU) enhance the network's capacity to learn complex relationships.
- ReLU prevents vanishing gradients, while Logistic and Tanh improve expressiveness, leading to higher performance.

AdaBoost Technique



1. Base Estimator:

- The base estimator for AdaBoost is a **Decision Tree** with a maximum depth of 1, making it a **stump**. This helps in focusing on learning weak classifiers to improve overall performance.

2. Number of Estimators:

- The AdaBoost classifier is initialized with `n_estimators=100`. This defines the number of boosting rounds or weak learners to be combined.

3. Random State:

- A fixed `random_state` ensures reproducibility.

4. Training:

- The balanced and scaled training data is used to train the AdaBoost model.

5. Feature Importance:

- The importance of each feature is computed and visualized, helping in identifying the key contributors to the model's performance.

6. Training and Validation Error Curves:

- Errors on the training and validation datasets are plotted as the number of estimators increases. This helps in understanding the behavior of the model and potential overfitting or underfitting trends.

Ada-boost for features analysis



- **Focus on Important Features:**

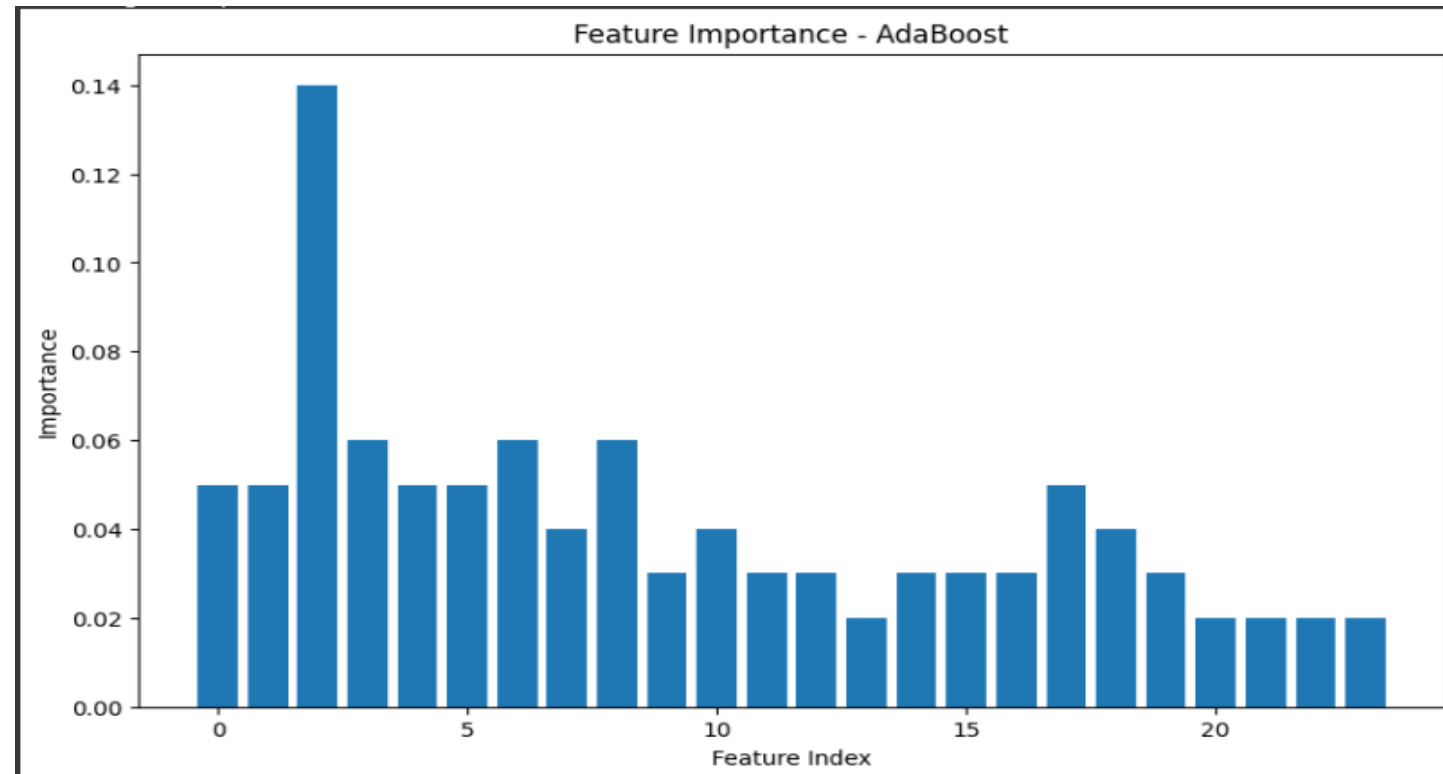
These features likely capture critical information related to the target variable (e.g., Chronic Medical Conditions).

- **Dimensionality Reduction:**

Features with consistently low importance could be removed or combined with other features to reduce computational cost and potential noise.

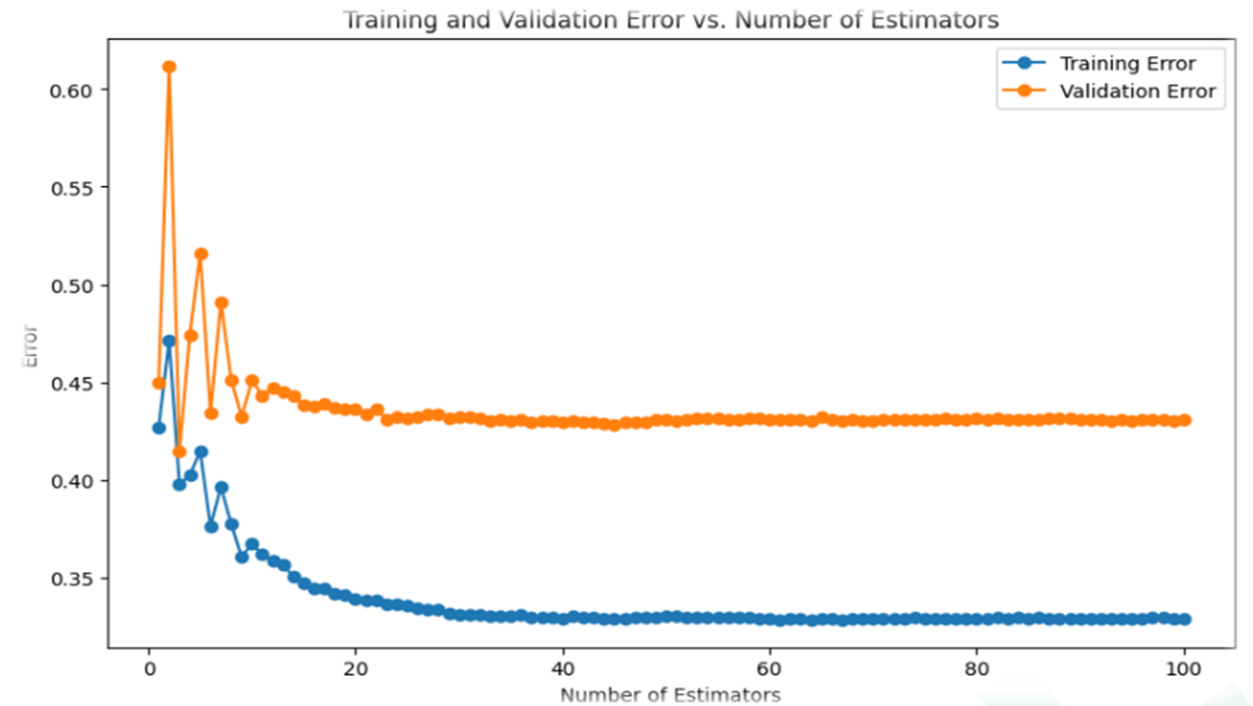


Ada boost



- Understanding feature importance helps interpret the model's behavior and ensures its predictions are based on meaningful attributes, which is crucial for a healthcare-related dataset like depression data.

Ada-boost
Accuracy–
56.91%



Training and validation error in Ada-boost

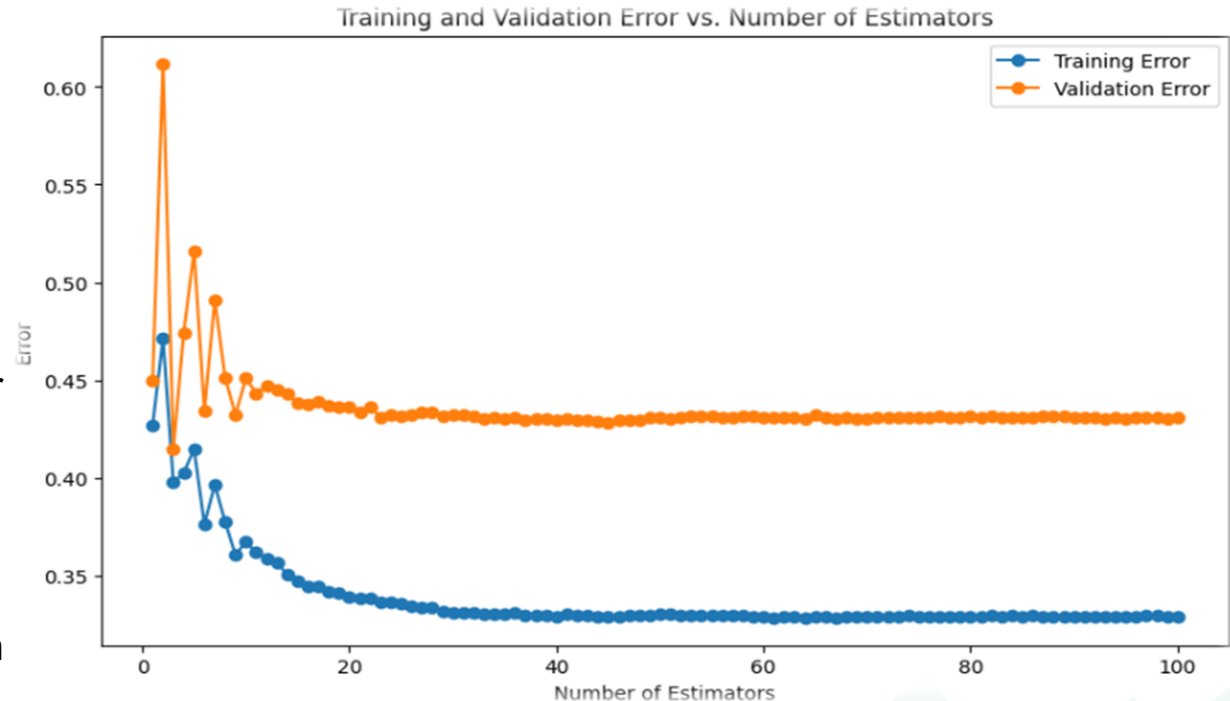


1. Training Error:

- The blue curve represents the training error, which consistently decreases as the number of estimators increases.
- This indicates that as more weak learners (decision stumps) are added, the model becomes increasingly adept at capturing patterns in the training dataset.
- However, after around **40 estimators**, the training error plateaus, meaning additional estimators no longer significantly improve performance on the training data.

2. Validation Error:

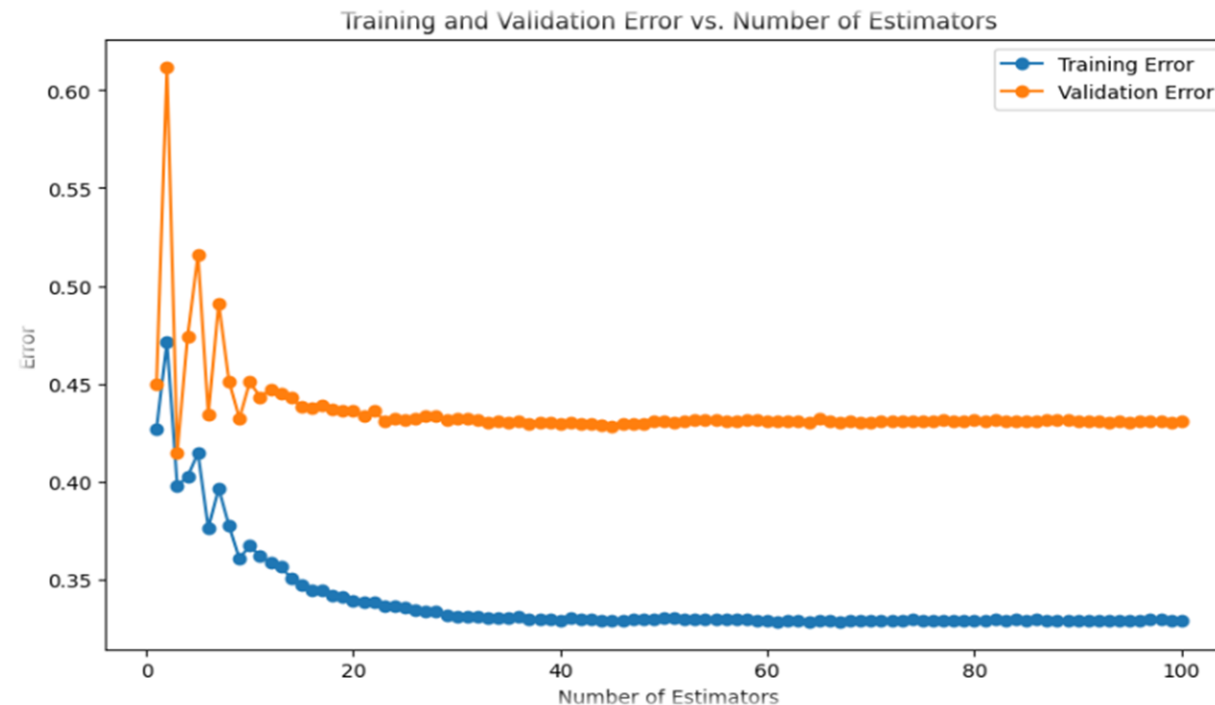
- The orange curve represents the validation error, which initially fluctuates but stabilizes after approximately **20 estimators**.
- The consistent gap between training and validation error suggests the model generalizes reasonably well, although some overfitting might occur if the gap widens with more estimators.



Convergence –:



Both training and validation errors stabilize around the **30-40 estimator** mark. This implies that increasing the number of estimators beyond this point may not yield significant improvements and could lead to computational inefficiency.



Model Ensemble Approach: Voting Classifier



The code implements **ensemble learning** by combining the predictions of three individual models:

Logistic Regression

Decision Tree Classifier

Gaussian Naive Bayes

Voting Classifier: Combines predictions from these base models using soft voting, which averages the predicted probabilities to make a final decision. This often improves overall model performance by leveraging the strengths of each base model.

- **Soft Voting:** Averages the class probabilities predicted by individual models, which typically provides better performance than hard voting (based on majority voting).
- This approach ensures that models with high confidence in their predictions have a stronger influence on the final decision.



Hyperparameters of Voting classifier



Logistic Regression: `class_weight='balanced'` accounts for class imbalance, and `max_iter=1000` ensures sufficient iterations for convergence.

Decision Tree: `max_depth=5` limits the tree's depth, controlling overfitting and complexity.

Gaussian Naive Bayes: Assumes a Gaussian distribution for continuous features.



VOTING CLASSIFIER

ACCURACY: **60.10%**

```
Confusion Matrix:
[[32996  8638]
 [16128  4303]]
Classification Report:
```

	precision	recall	f1-score	support
False	0.67	0.79	0.73	41634
True	0.33	0.21	0.26	20431
accuracy			0.60	62065
macro avg	0.50	0.50	0.49	62065
weighted avg	0.56	0.60	0.57	62065

Confusion Matrix of voting classifier

Training & Validation Accuracy for Voting Classifier

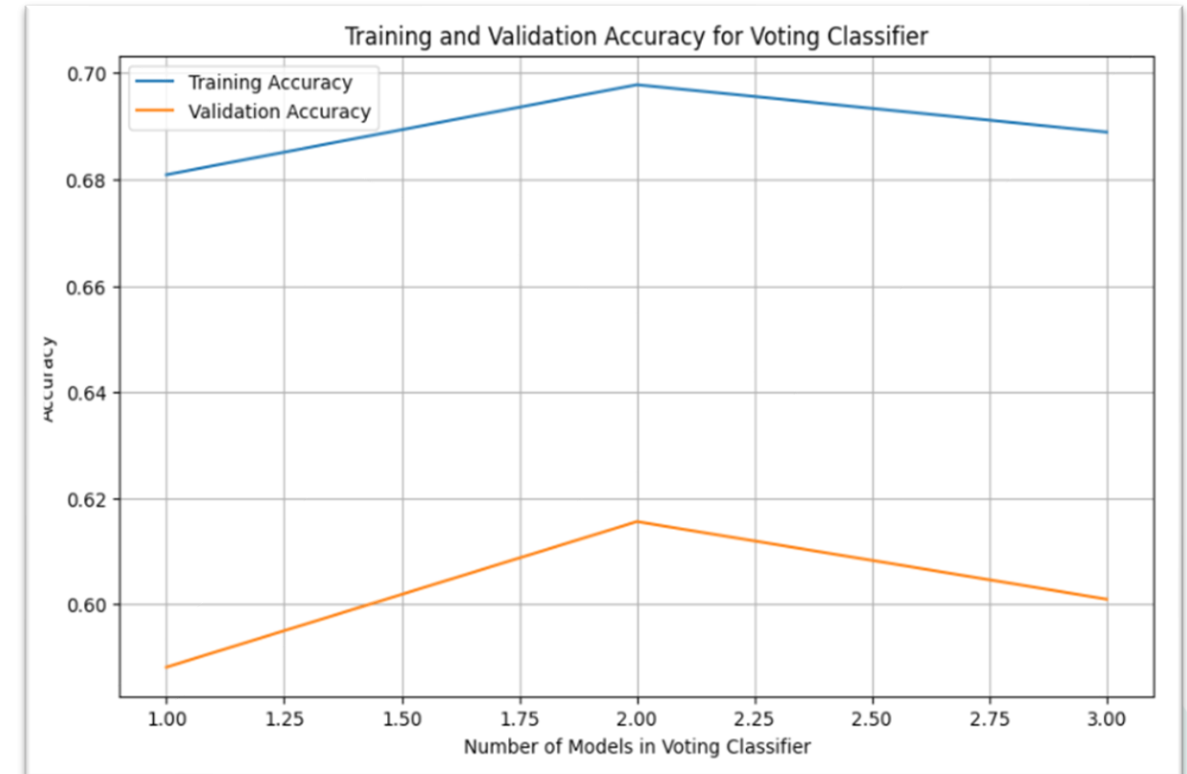


1. Training Accuracy:

- The training accuracy increases as more models are added to the Voting Classifier, peaking when all three models are included.
- After reaching the maximum accuracy with two models, there is a slight drop when the third model is included. This may indicate overfitting or conflicting model predictions in the ensemble.

2. Validation Accuracy:

- The validation accuracy follows a similar trend, increasing with the addition of the first two models but then slightly declining when the third model is included.
- The peak validation accuracy is observed when **two models** are included in the ensemble, likely indicating the optimal combination of diverse models that generalize well.



XGBoost Classifier Initialization:



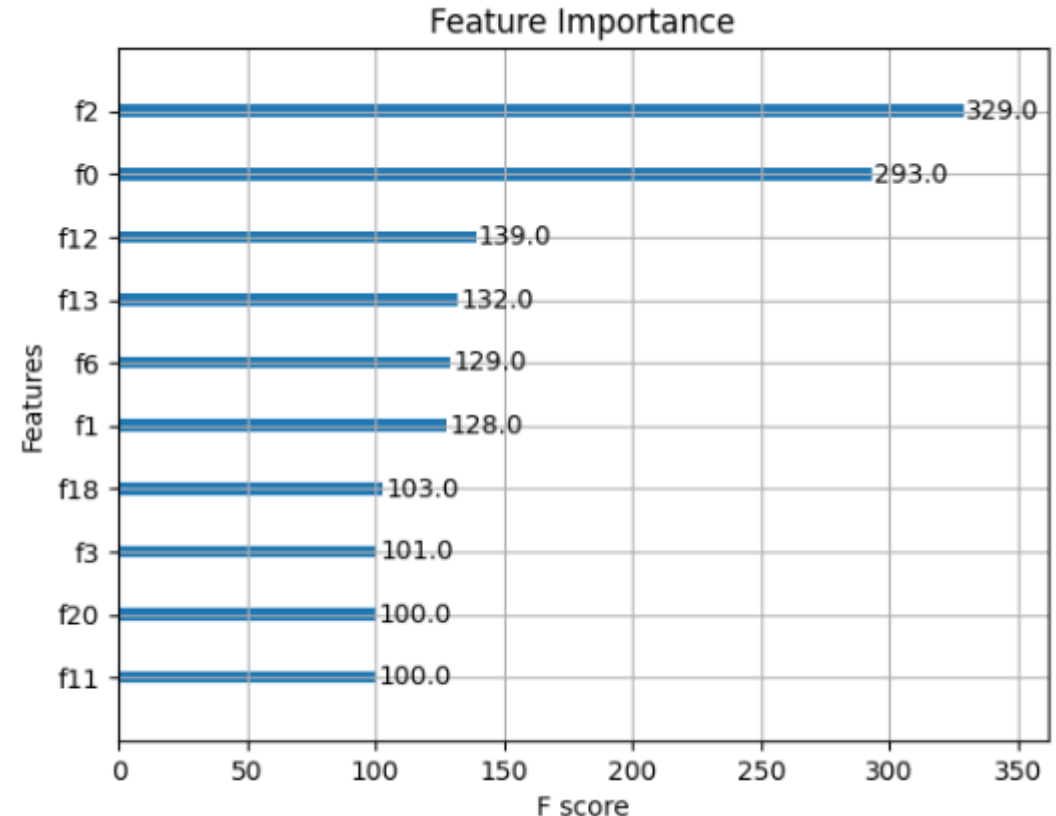
1. **n_estimators=200**: Specifies 200 boosting rounds (trees) to build the ensemble.
2. **max_depth=4**: Limits the depth of each decision tree to prevent overfitting while capturing complex patterns.
3. **learning_rate=0.1**: Determines the step size for adjusting weights to reduce errors.
4. **colsample_bytree=0.8** and **subsample=0.8**: Controls the fraction of columns and rows sampled for each tree, adding randomness to enhance generalization.
5. **eval_metric="logloss"**: Optimizes the logarithmic loss function to handle imbalanced datasets effectively.
6. **random_state=42**: Ensures reproducibility.

Analysis of the Feature Importance Plot



Top Contributing Features:

- **f2** and **f0** are the most influential features, with F-scores of 329 and 293, respectively. These features are used most frequently for decision splits in the trees.
- This indicates that the dataset's predictive power is highly concentrated in these features.

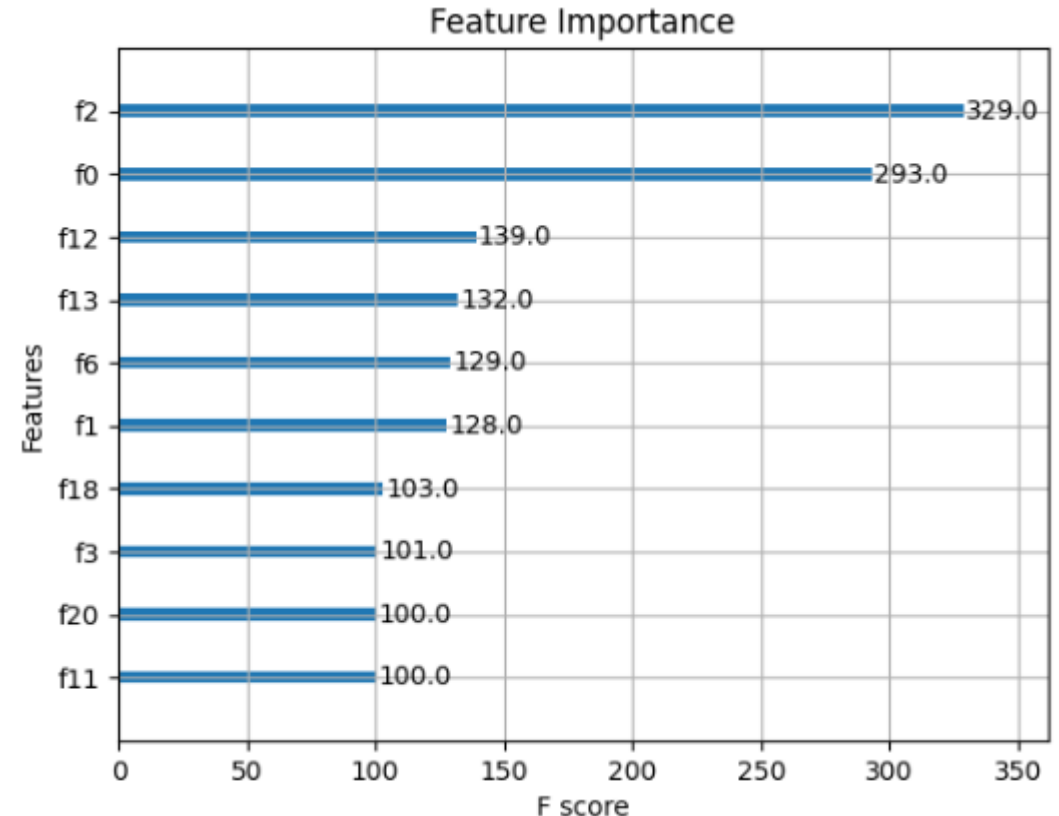


Analysis of the Feature Importance Plot



Significant but Secondary Features:

- Features like **f12**, **f13**, **f6**, and **f1** have moderate importance scores (ranging from ~128 to ~139).
- These features contribute meaningfully to the model but are not as impactful as **f2** and **f0**.

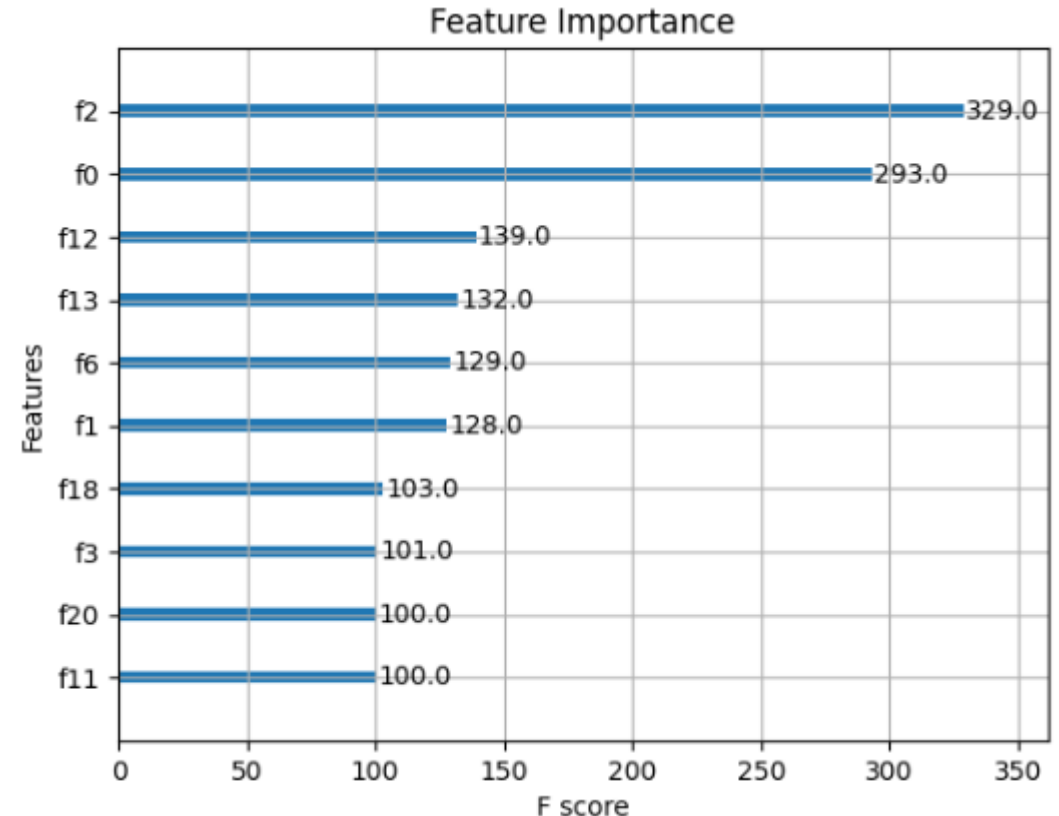


Analysis of the Feature Importance Plot



Low-Contributing Features:

- Features like **f18**, **f3**, **f20**, and **f11** show relatively low importance (scores ~100). While they still influence the model, their contribution is limited compared to the top features.



Analysis of the XGBoost Log Loss per Epoch Curve

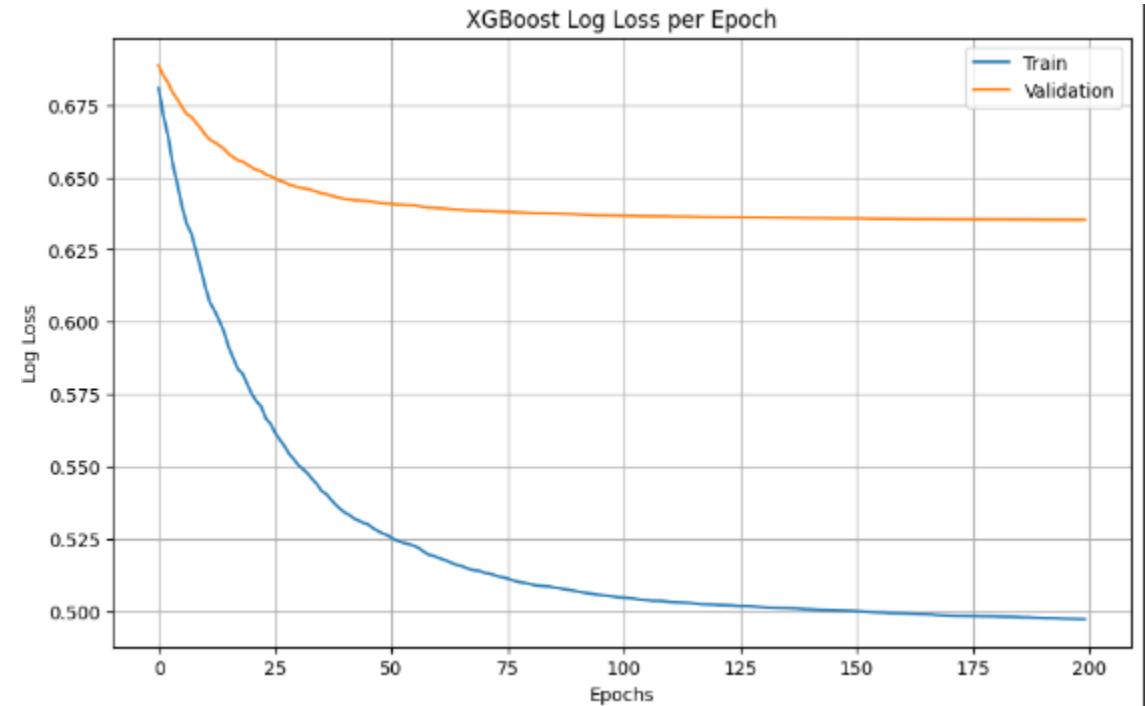


Observations:

- The **training log loss (blue curve)** decreases sharply during the initial epochs, indicating that the model is quickly learning patterns from the training data.
- The **validation log loss (orange curve)** decreases initially but then plateaus after about 50 epochs, showing that the model has reached its optimal performance on the validation set.

Model Performance:

- The steady decline in training log loss reflects the increasing accuracy of predictions on the training set.
- The plateauing of the validation log loss suggests that further training does not significantly improve generalization and might lead to overfitting if continued.



XGBOOST Ensemble Accuracy



XGBOOST Ensemble
ACCURACY-66.78%

Confusion Matrix:

```
[[41128  506]
```

```
[20112  319]]
```

Classification Report:

	precision	recall	f1-score	support
False	0.67	0.99	0.80	41634
True	0.39	0.02	0.03	20431
accuracy			0.67	62065
macro avg	0.53	0.50	0.41	62065
weighted avg	0.58	0.67	0.55	62065

Linear Discriminant Analysis (LDA)



Purpose: LDA is a classification technique that works by finding a linear combination of features that best separates the classes. It assumes that the data from each class has a Gaussian distribution with identical covariance matrices.



Linear Discriminant Analysis (LDA)



Linear Discriminant Analysis –

ACCURACY: 59.95%

```
Confusion Matrix (LDA):  
[[32952  8682]  
 [16173  4258]]  
Classification Report (LDA):  
              precision    recall  f1-score   support  
  
   False       0.67       0.79       0.73     41634  
    True       0.33       0.21       0.26     20431  
  
 accuracy              0.60     62065  
  macro avg              0.50     62065  
weighted avg              0.56     62065
```

Quadratic Discriminant Analysis (LDA)



Purpose: QDA is an extension of LDA that allows each class to have its own covariance matrix, making it suitable for problems with non-linear boundaries.

Key Feature: QDA introduces flexibility to model non-linear class separations, making it useful for more complex datasets.



Quadratic Discriminant Analysis (QDA)



Linear Discriminant Analysis –

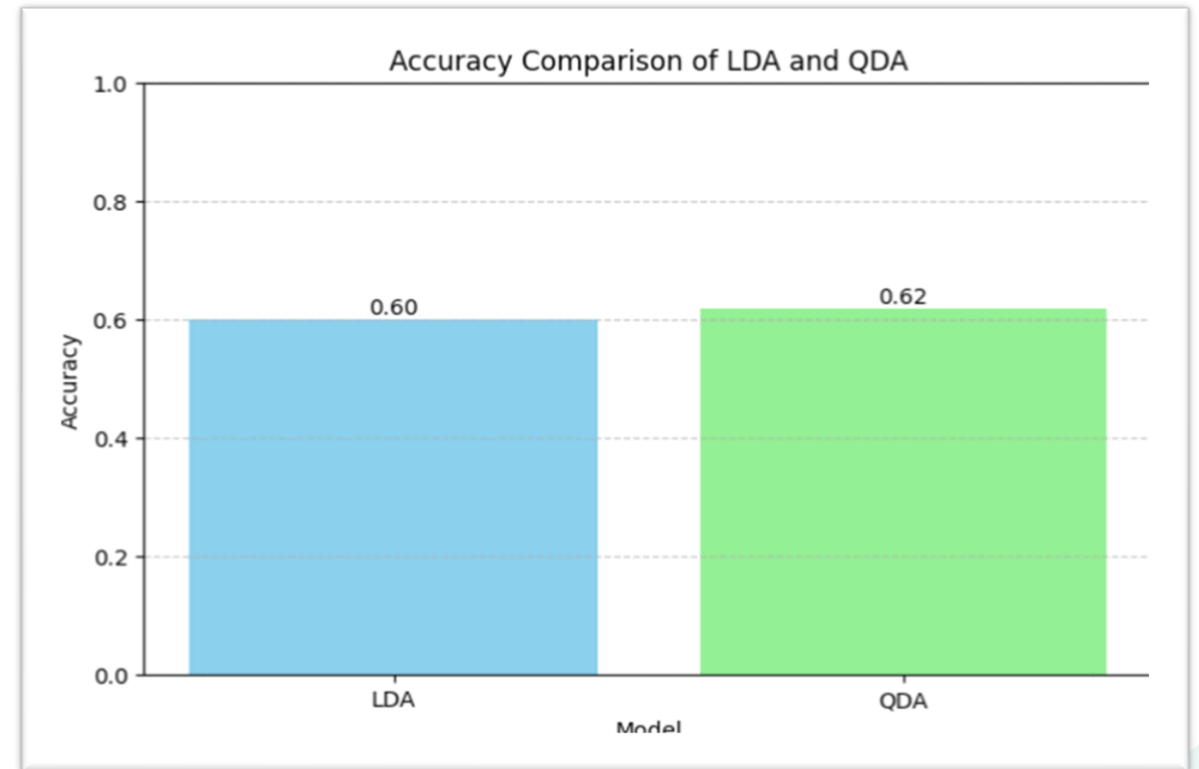
ACCURACY: 59.95%

```
Confusion Matrix (LDA):  
[[32952  8682]  
 [16173  4258]]  
Classification Report (LDA):  
              precision    recall  f1-score   support  
  
   False       0.67       0.79       0.73     41634  
    True       0.33       0.21       0.26     20431  
  
   accuracy              0.60     62065  
  macro avg       0.50       0.50       0.49     62065  
weighted avg       0.56       0.60       0.57     62065
```

LDA VS QDA [Accuracy]



- **LDA:** Works well when the dataset has linear separability and equal class covariances.
- **QDA:** Handles non-linear class boundaries but requires more data due to higher complexity.

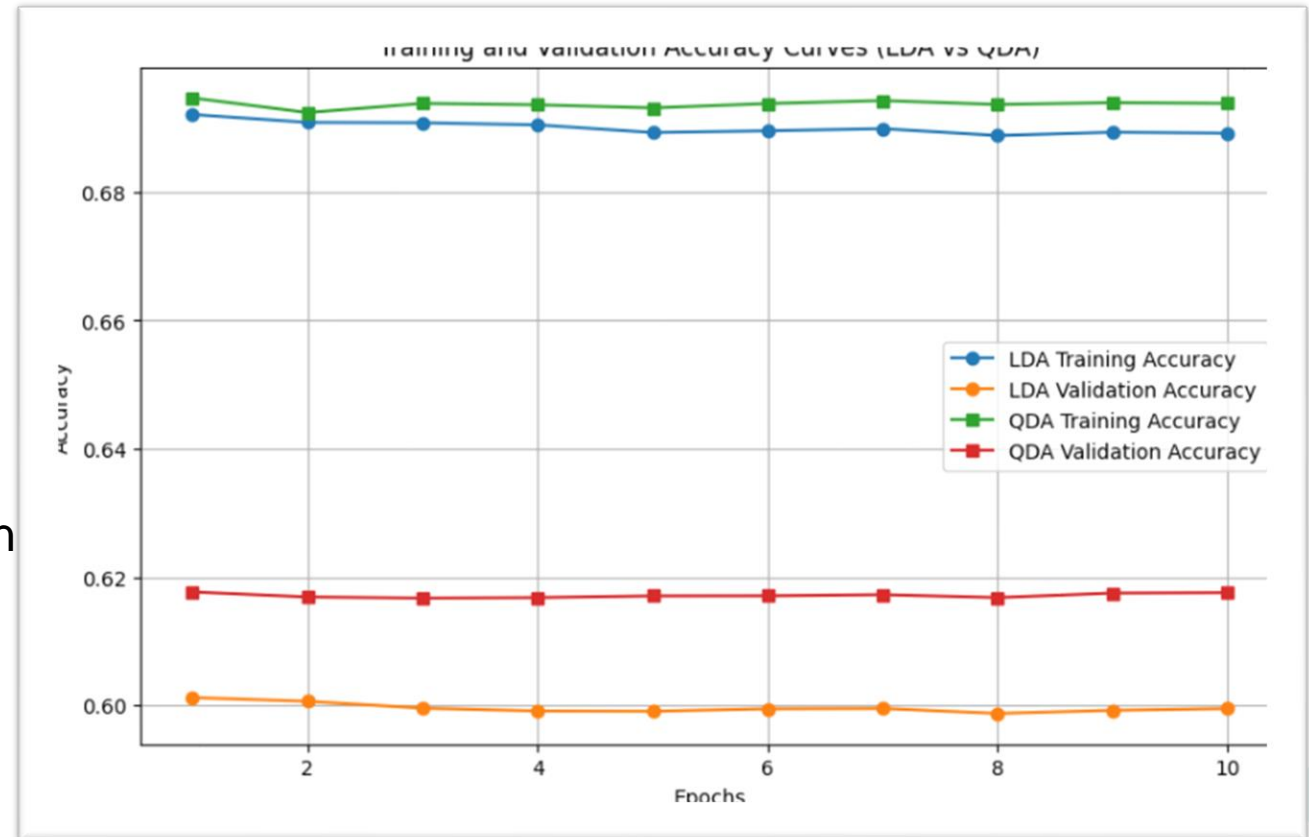


LDA Performance



Training Accuracy: The curve for LDA training accuracy is consistently high, hovering around 68.5%. This indicates that LDA is performing well on the training data.

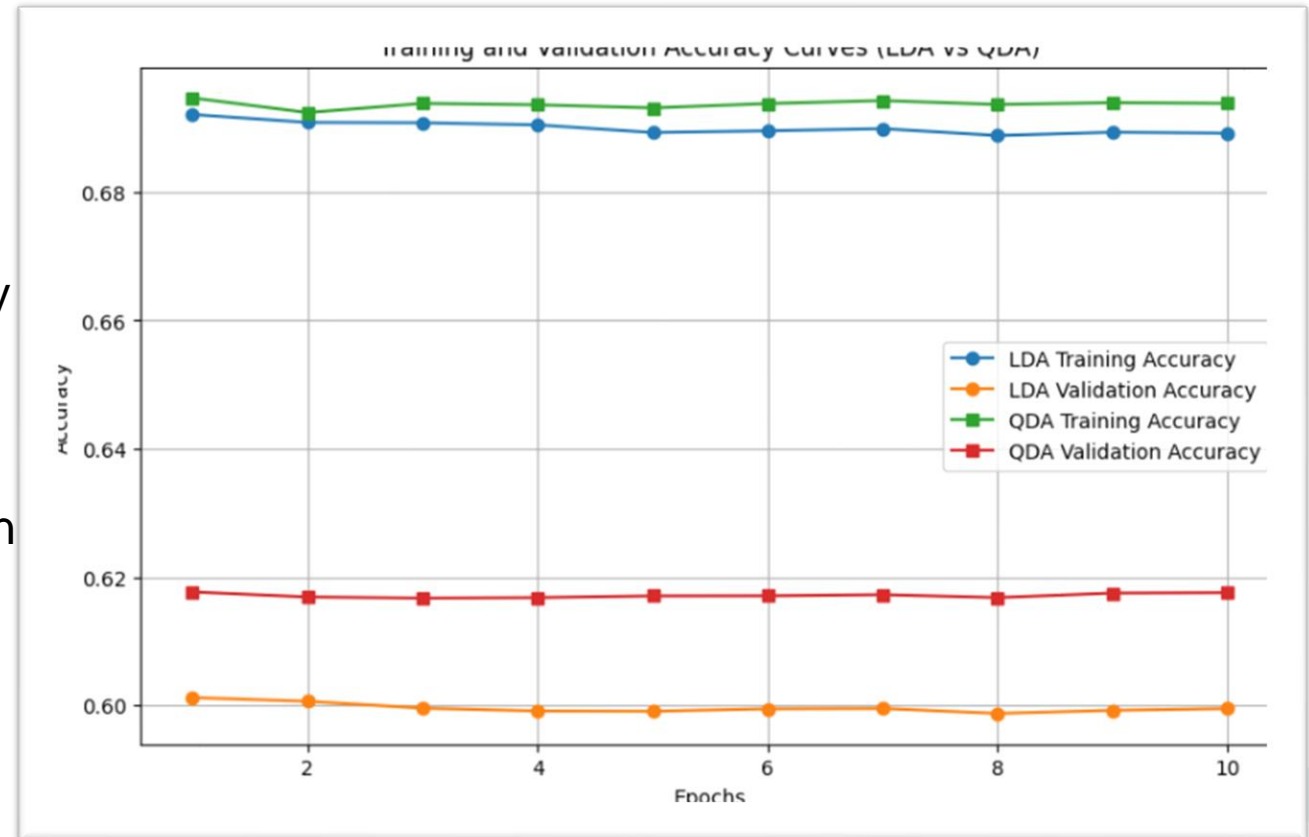
Validation Accuracy: The validation accuracy is slightly lower, staying around 60%. The gap between training and validation accuracy suggests that LDA might slightly overfit the training data but still generalizes decently.



QDA Performance



- **Training Accuracy:** QDA training accuracy is consistent and lower than LDA, at around 62%. This reflects that QDA is less capable of fitting the training data, likely due to its increased complexity and the requirement for larger datasets.
- **Validation Accuracy:** The QDA validation accuracy remains steady around 60%, similar to its training accuracy. This suggests that QDA is less prone to overfitting compared to LDA but may struggle to capture the data's structure effectively.



Results (Midsem)



Machine Learning Technique	Accuracy(%)
Logistic Regression with Batch Gradient Descent (BGD)	57
Logistic Regression with Stochastic Gradient Descent (SGD)	52
L2 (Ridge) Logistic Regression with Recursive Feature Elimination (RFE)	57
Logistic Regression with PCA	54
Logistic Regression with SMOTE and OPTIMAL THRESHOLD	60
Random Forest (Default parameters)	59
Random Forest (Tuned)	61

Results (Midsem)

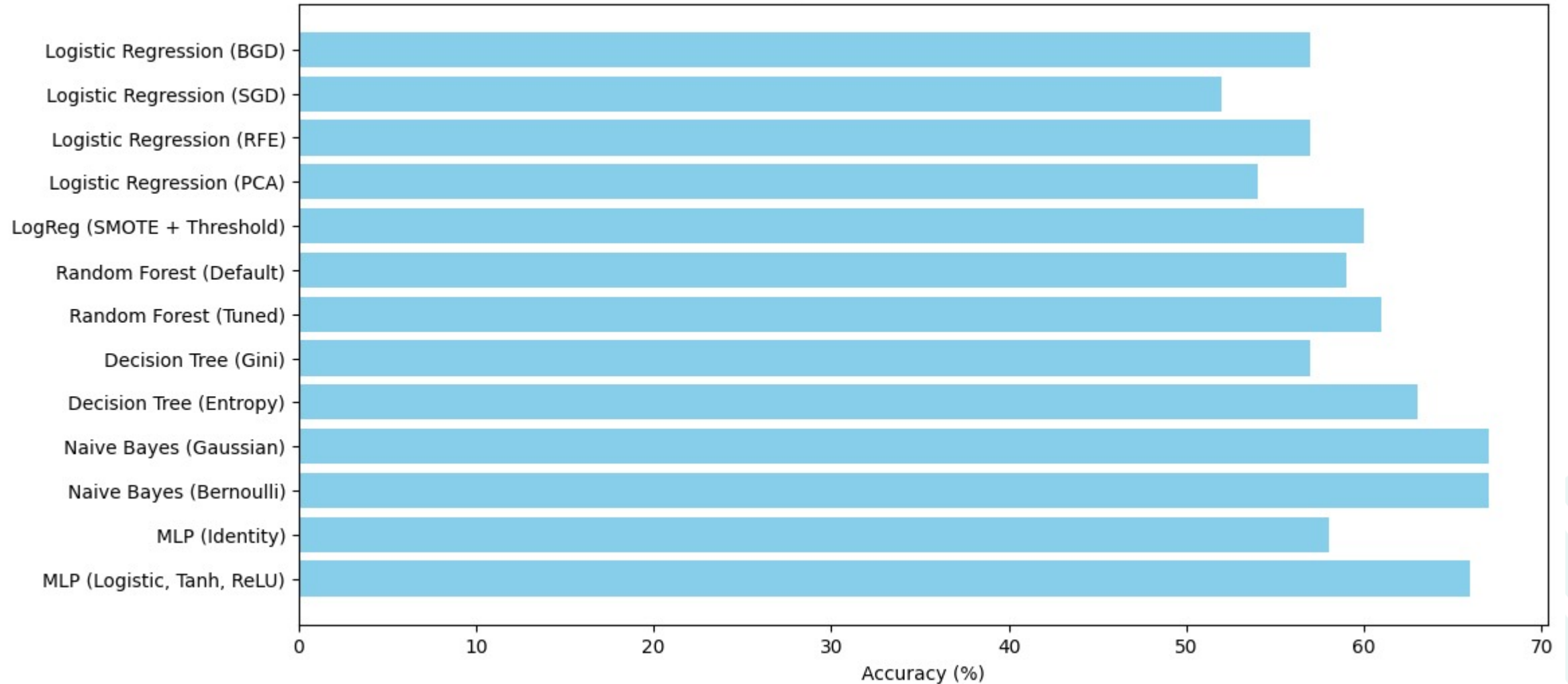


Decision Tree Classifier [CRITERIAN='GINI']	57
Decision Tree Classifier [CRITERIAN='ENTROPY']	63
Naive Bayes (Gaussian)	67
Naive Bayes (Bernoulli)	67
MLP Classifier('IDENTITY')	58
MLP Classifier('LOGISTIC','TANH','RELU')	66

Results (Midsem)



Comparison of Model Accuracies



Logistic Regression with SMOTE and Optimal Threshold vs. Basic Logistic Regression:

- Logistic Regression with SMOTE and threshold tuning (60%) outperforms the basic version (57%) by addressing class imbalance and refining decision boundaries, improving the separation between classes.

Tuned Random Forest vs. Default Random Forest:

- The tuned Random Forest (61%) performs better than the default version (59%) due to optimized hyperparameters (like `max_depth` and `n_estimators`), reducing overfitting and improving generalization..

Decision Tree (Entropy) vs. Decision Tree (Gini):

- The entropy-based Decision Tree (63%) achieves higher accuracy compared to Gini (57%) because it captures finer information gains during splits, enhancing class separation.

Analysis – Machine Learning Models



- **4) Naive Bayes vs. Logistic Regression:**
 - Naive Bayes (67%) performs better than Logistic Regression (57%) by efficiently handling the dataset's feature independence and class imbalance, making it well-suited for binary classification tasks.
- **5) MLP with Non-linear Activation vs. Identity Activation:**
 - MLP with non-linear activations (66-67%) outperforms identity activation (58%) as nonlinear functions like ReLU, Tanh, and Logistic allow the network to learn complex patterns, improving classification performance.



Comparison Analysis -

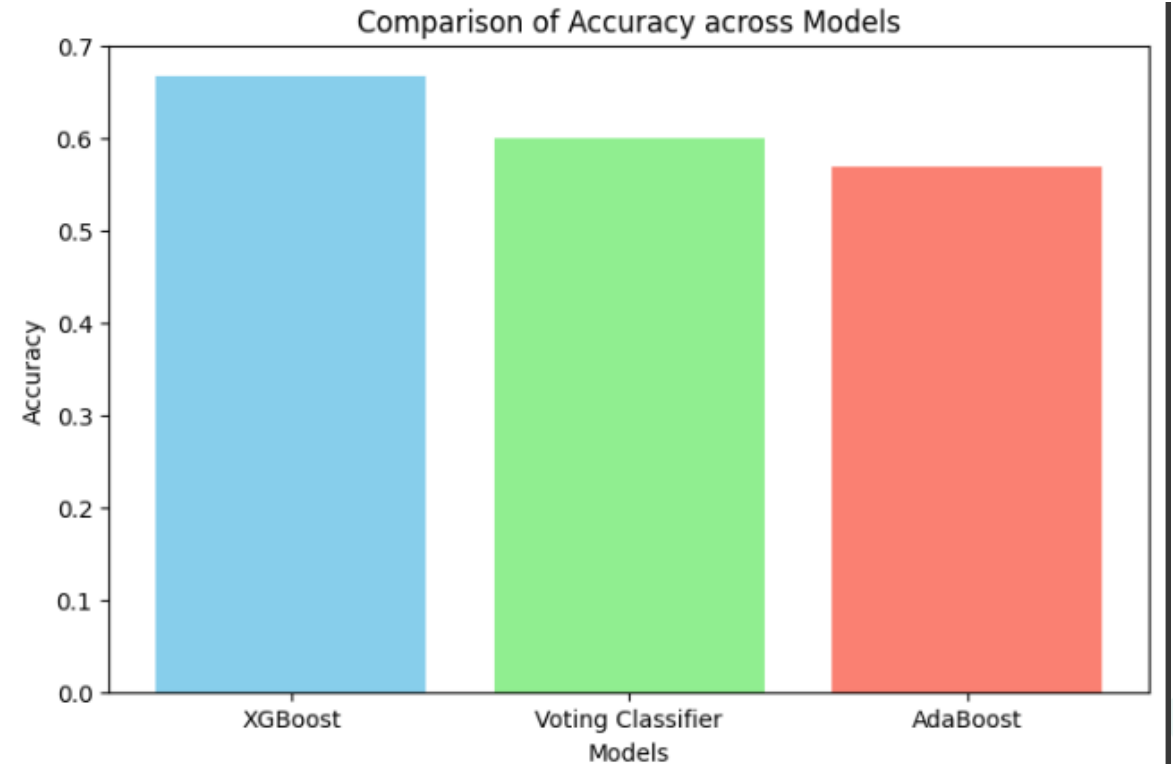


1. Voting Classifier:

- Accuracy: **60.1%**
- Strengths: Combines predictions from multiple models (Logistic Regression, Decision Tree, and Naive Bayes) using soft voting, leveraging their strengths.

2. Linear Discriminant Analysis (LDA):

- Accuracy: **59.95%**
- Strengths: Performs well with linearly separable data and balances precision and recall fairly well.



Comparison Analysis –



1. Quadratic Discriminant Analysis (QDA):

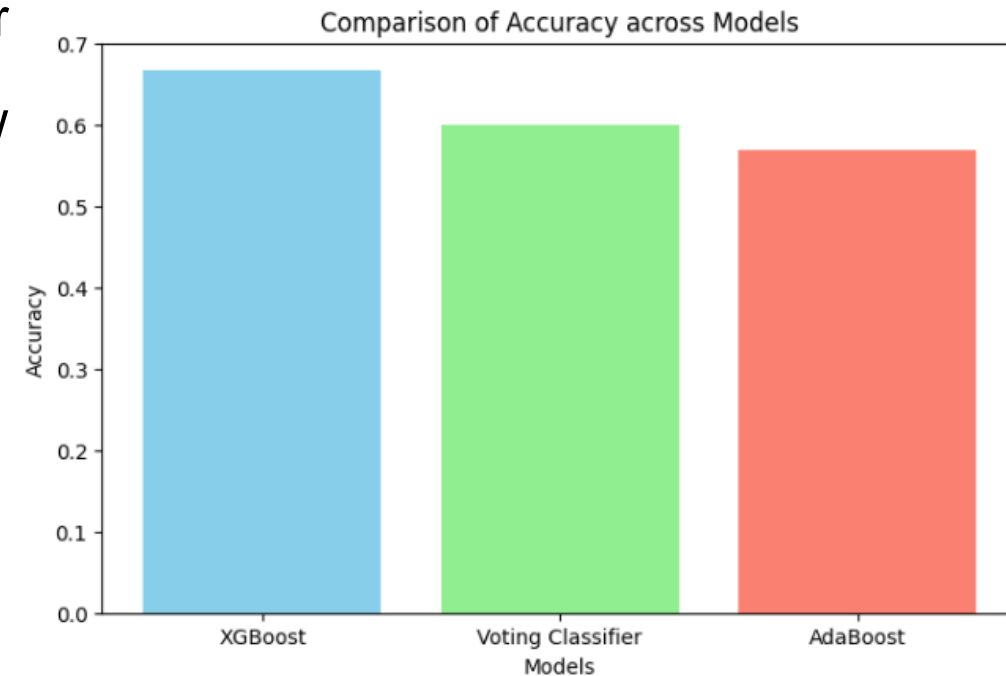
- Accuracy: **61.8%**
- Strengths: Outperforms LDA and the Voting Classifier by capturing some non-linear relationships. Precision is higher compared to LDA, though recall remains low

2. XGBoost:

- Accuracy: **66.8%**
- Strengths: Gradient boosting with optimized decision trees leads to better performance than other techniques. Particularly excels in leveraging relationships across features for improved accuracy.

3. AdaBoost:

- Accuracy: **56.9%**
- Strengths: Boosts weak learners iteratively, showing decent recall compared to other methods like XGBoost.



Best Performing Technique: XGBoost (Accuracy: 66.8%)



1. **Gradient Boosting Strengths:** XGBoost effectively handles large, high-dimensional datasets and learns complex relationships through sequential boosting.
2. **Hyperparameter Optimization:** Tuning parameters such as learning rate, max depth, and subsampling allows for a better fit to the data.
3. **Feature Importance:** XGBoost effectively identifies and utilizes the most significant features for classification.
4. **Adaptability:** Handles missing values and categorical encoding better than techniques like LDA or QDA.

Conclusion



- Naive Bayes (67%) performed best due to the dataset's alignment with the independence assumption and SMOTE balancing.
- Decision Tree (63%) captured non-linear patterns with well-tuned hyperparameters, while Random Forest (61%) handled feature interactions but struggled with variance. MLP (66-67%) utilized non-linear activations to learn complex patterns effectively.
- Logistic Regression with SMOTE and threshold tuning (60%) improved classification but fell short compared to non-linear models.

Overall, tuning, regularization, and class balancing were key to optimizing performance.



Why MLP with Non-linear Activation Functions and XGBoost are the Best Performing Models:



Conclusion

Why MLP with Non-linear Activation Functions and XGBoost are the Best Performing Models:

Handling Non-linear Relationships:

Both models excel at learning non-linear relationships and feature interactions, which are common in real-world datasets like this one.

Feature Importance:

XGBoost's feature importance mechanism and MLP's ability to learn hierarchical patterns allowed them to leverage the most impactful features effectively.

Regularization:

Both models have in-built mechanisms to prevent overfitting (e.g., early stopping in MLP, regularization in XGBoost), ensuring robust performance on unseen data.

Balancing Techniques:

The use of SMOTE ensured that both models focused equally on minority and majority classes, enhancing recall and precision.

MLP with Non-linear Activation Functions (67% Accuracy):

- **Non-linearity in Activation Functions:**
 - Non-linear activation functions like `relu`, `tanh`, and `logistic` enable the model to learn complex relationships in the data that linear models cannot capture.
 - `ReLU` specifically helps in faster convergence by avoiding the vanishing gradient problem and creating sparsity in activations.
- **Multi-layer Architecture:**
 - The two hidden layers (100 and 50 neurons) allow MLP to learn hierarchical representations of features, capturing both low-level and high-level patterns in the data.
- **Adaptive Optimization (Adam):**
 - The `adam` optimizer dynamically adjusts the learning rate, leading to faster convergence and robustness across different activation functions and datasets.
- **Versatility with Features:**
 - MLP is capable of learning non-linear relationships in feature interactions, which is particularly useful in complex datasets like this one, where dependencies between features may not be linear.

XGBoost (67% Accuracy):

- **Boosting Framework:**
 - XGBoost uses **gradient boosting**, an iterative process where each tree corrects the errors of the previous ones, leading to a highly accurate ensemble of decision trees.
- **Feature Importance Utilization:**
 - XGBoost assigns feature importance scores during training, effectively focusing on the most impactful features, reducing noise and enhancing predictive power.
- **Regularization and Pruning:**
 - Regularization parameters prevent overfitting, while the max-depth of 4 and learning rate of 0.1 ensure that the trees are shallow enough to generalize well while learning fine-grained patterns.
- **Column Subsampling and Row Subsampling:**
 - By using `colsample_bytree=0.8` and `subsample=0.8`, XGBoost reduces variance and avoids overfitting by training trees on random subsets of features and data points.

Future Works



- In future, we will apply SVM for better boundaries and ensemble methods like AdaBoost, XGBoost, and Voting Classifier to enhance accuracy and reduce bias. These techniques will help optimize predictions and handle complex patterns effectively.

DONE ALL WORKS !!!



Individual Member Contributions



All the members contributed equally to the work and helped each other in making edits to the codes and writing this report. The individual contributions listed below are only representations of the assignments of tasks to each team member.

- **Satyam:** Decision Tree, MLP
- **Aditya:** Logistic Regression ,Decision Tree
- **Suyash:** Random Forest, Model Evaluation
- **Vickey:** Data Preprocessing , Naive Bayes
- Satyam Pandey: Documentation and EDA



THANK YOU!

